Circuit and Method for Service Clock Recovery

Cross Reference to Related Cases

[0001] This application is related to commonly-assigned US Patent No. 6,157,646, entitled "CIRCUIT AND METHOD FOR SERVICE CLOCK RECOVERY," and issued December 5, 2000 (the '646 Patent.) The '646 Patent is incorporated herein by reference.

[0002] This application is also related to commonly-assigned, co-pending US Application Serial No. 09/443,662, entitled "ADAPTIVE CLOCK RECOVERY FOR CIRCUIT EMULATION SERVICE" (the '662 Application) filed on November 19, 1999 (pending).

Technical Field of the Invention

[0003] The present invention relates generally to the field of telecommunications and, in particular, to a circuit and method for service clock recovery.

Background of the Invention

[0004] Asynchronous Transfer Mode (ATM) is a packet oriented technology which permits continuous bit rate signals carrying one or more of voice, video, and data, to be conveyed across a broadband network within packets. ATM is suitable for the transport of bursty traffic such as data, as well as accommodating constant or continuous bit rate signals. In delivering continuous bit rate traffic (e.g., T1, DS3 signals) in a broadband network, a service clock controlling a destination node buffer must operate at a frequency precisely matched to that of a service clock at a source node in order to avoid buffer overflow or underflow and resulting loss of data.

[0005] One problem with synchronizing the service clock at the destination node with the service clock at the source node is "cell jitter" that is inherent in the use of an ATM network. Cell jitter is the random delay and aperiodic arrival of cells at a destination node. In other words, cell jitter means that all of the cells or packets that travel between a source node and a destination node do not take the same amount of

time to travel through the ATM network. Thus, it is difficult to use the cell arrival times to directly synchronize the frequency of the service clock at the destination node with the service clock at the source node.

[0006] Numerous schemes have been proposed to provide a mechanism for synchronizing the service clocks of source and destination nodes in the presence of cell jitter. This is also referred to as "service clock recovery." Descriptions of many of these schemes are provided in Fleischer et al. U.S. Pat. No. 5,260,978 (the '978 Patent). The '978 Patent incorporated by reference. Perhaps the most elegant and widely accepted of the service clock recovery schemes is known as synchronous residual time stamp (SRTS or RTS) encoding. SRTS encoding is specified, for example, in T1.630, Annex A for delivering T1 service over an ATM network.

[0007] The '978 Patent describes one embodiment of SRTS encoding. A free running four bit counter is used at the source node to count cycles of a common network clock. At the end of every residual time stamp (RTS) time period formed by 3008 service clock cycles (i.e., eight cells of forty-seven bytes of data each), the current four bit count of the four bit counter is transmitted in the ATM adaptation layer (AAL1) by using one bit in every other byte of the AAL1 for eight cells. It should be noted that the AAL1 is the overhead byte which accompanies the forty-seven bytes of data to constitute the forty-eight-byte payload of an ATM cell. The ATM cell also includes five additional bytes of header. The four-bit SRTS provides sufficient information for unambiguously representing the number of network clock cycles within a predetermined range.

[0008] The clock recovery at the destination node according to the '978 Patent involves determining from the received RTSs the number of network clock cycles in each RTS period, and generating a pulse signal from the network clock in which the periods are determined by the number of network clocks represented by the received RTSs . The pulse frequency is then multiplied by 3008 in order to recover the source node service clock.

[0009] While the clock recovery mechanism of the '978 Patent might be suitable for recovering the source node service clock, it is neither the only recovery mechanism

possible, nor necessarily the most optimal mechanism for recovering the source node service clock.

[0010] U.S. Patent No. 5,608,731, entitled *Closed Loop Clock Recovery for Synchronous Residual Time Stamp* (the '731 Patent) describes another service clock recovery technique using SRTS. The '731 Patent describes an apparatus with a digitally controlled oscillator at the destination node. A local RTS-related value is generated at the destination node based on the output of the digitally controlled oscillator. RTS values from incoming data packets are compared to the local RTS-related values generated at the destination node. This provides a feedback error or control signal. The control signal is used to adjust the digitally controlled oscillator at the destination node. With the feedback loop as provided, when the destination node clock is faster than the source clock, the error signal will cause the destination node clock to slow, and vice versa. Unfortunately, the complex circuitry used in this closed loop must be replicated for each port at the destination node. Further, this closed loop solution suffers from problems common to closed loop control circuits.

[0011] For the reasons stated above, and for other reasons stated below which will become apparent to those skilled in the art upon reading and understanding the present specification, there is a need in the art for an improved circuit and method for service clock recovery.

Summary of the Invention

[0012] The above mentioned problems with service clock recovery in a telecommunications network using asynchronous transfer mode and other problems are addressed by the present invention and will be understood by reading and studying the following specification. Service clock recovery is described using a direct digital synthesis (DDS) circuit that is set based on control values calculated over a plurality of time periods. This technique is applicable to a variety of clock recovery mechanisms including but not limited to mechanisms using Residual Time Stamp (RTS) values and adaptive clock recovery using buffer fill levels at the destination node .

[0013] In particular, one embodiment of the present invention provides a method for synchronizing a service clock at a destination node with a service clock at a source node for circuit emulation service over a packet network. The method includes receiving data packets from a source node at at least one port of the destination node. At the destination node, the method removes from the data packets residual time stamp (RTS) values that were created at the source node based on at least the service clock at the source node. The method further determines a majority count and a minority count of RTS values over a plurality of time periods. The method uses the majority and minority counts for the plurality of time periods to set the frequency of a service clock at the destination node for use in receiving data packets.

Brief Description of the Drawings

[0014] Figure 1 is a block diagram of an embodiment of a service clock recovery circuit constructed according to the teachings of the present invention.

[0015] Figure 2 is a flow chart of an embodiment of a process for setting a frequency of a local service clock according to the teachings of the present invention.

[0016] Figure 3 is a block diagram of an embodiment of a packet network that uses adaptive clock recovery at a destination node according to the teachings of the present invention.

Detailed Description of the Invention

[0017] In the following detailed description of the preferred embodiments, reference is made to the accompanying drawings which form a part hereof, and in which is shown by way of illustration specific illustrative embodiments in which the invention may be practiced. These embodiments are described in sufficient detail to enable those skilled in the art to practice the invention, and it is to be understood that other embodiments may be utilized and that logical, mechanical and electrical changes may be made without departing from the spirit and scope of the present invention. The following detailed description is, therefore, not to be taken in a limiting sense.

I. Overview

[0018] Embodiments of the present invention provide improvements in techniques for recovering a service clock at a destination node. A first embodiment, described in section I below, relates to recovery of a service clock using residual time stamp (RTS) values. The second embodiment describes a system that uses an adaptive clock recovery technique to recover the service clock. A common theme in both embodiments is that the service clock is controlled based on values calculated over a plurality of time periods.

II. Service Clock Recovery with Synchronous Residual Time Stamp

[0019] Figure 1 is a block diagram of an embodiment of a circuit for recovering a service clock at destination node 100 according to the teachings of the present invention. Destination node 100 receives data packets from source node 104 over packet network 102. Source node 104 includes a service clock that is used to clock data packets for circuit emulation service over network 102. In order to properly process the packets received at destination node 100, destination node 100 needs to "recover" the service clock of source node 104. In other words, destination node 100 needs to locally generate a service clock that is substantially synchronized with the service clock of source node 104 to provide acceptable circuit emulation service.

[0020] For simplicity in describing the techniques for service clock recovery, destination node 100 and source node 104 are generally described in the context of a single channel with transmission in a single direction. However, it is understood that both destination node 100 and source node 104 are bi-directional, multi-channel nodes. Thus, in one embodiment, each node of the network includes circuitry of the type described below that is used to synchronize a local service clock for each channel with a service clock at a remote node.

[0021] Source node 104 incorporates a signal into the data packets sent to destination node 100 that is used to recover the service clock at destination node 100. For example, source node 104 incorporates "residual time stamp" (RTS) values into the ATM Adaptation Layer of the data packets it transmits as described above. One

embodiment of this technique is described in U.S. Patent No. 5,608,731 entitled *Closed Loop Clock Recovery for Synchronous Residual Time Stamp* (the '731 Patent). The portion of the '731 Patent that describes the generation of RTS values (including Figure 1 and Col. 2, line 65 to Col. 3, line 36 of the '731 Patent), is incorporated by reference.

[0022] Destination node 100 includes circuitry that is used to remove and process the RTS values to generate a local service clock signal that is synchronized with the service clock at source node 104. ATM disassembler 108 is coupled to receive packets from packet network 102. ATM disassembler 108 disassembles the packets and passes the disassemble packets on to AAL overhead processor 110. AAL overhead processor 110 extracts RTS values and passes the RTS values to counting circuit 112. Further, AAL overhead processor 110 also passes data from the packets to buffer 114. Buffer 114 and counting circuit 112 are both coupled to microcontroller 116.

[0023] In one embodiment, microcontroller 116 uses information from counting circuit 112 to control direct digital synthesis circuit (DDS) 120 to generate a local service clock signal for destination node 100. In another embodiment, microcontroller 116 also uses a fill level of buffer 114 to control the frequency of the local service clock generated by DDS circuit 120.

[0024] Direct digital synthesis circuit 120 comprises a circuit that synthesizes an output sine wave at a frequency that is a fraction of a reference clock or oscillator, e.g., network clock 106. In one embodiment, network clock 106 is the OC-3 byte clock of a packet network. Typically, the output sine wave of direct digital synthesis circuit 120 has a frequency that is less than one-third of the frequency of the reference clock. Conventionally, the fraction used by a direct digital synthesis circuit is established using a large digital number, e.g, a 32 bit number. This results in a very high resolution on the frequency of the output sine wave. In other words, a direct digital synthesis circuit provides precise control over the frequency of its output signal. It is noted that the frequency stability of a direct digital synthesis circuit is the same as its reference clock. Thus, temperature and aging do not affect its output frequency as would happen with a digitally controlled oscillator of the '731 Patent.

[0025] For a direct digital synthesis circuit using an OC-3 byte clock and a 32 bit register, the output clock signal can be set to have a frequency within a tolerance of 5 parts per billion (PPB). The required frequency range for the service clock is ±200 parts per million (PPM), and for T1 the typical four bit residual time stamp (RTS) count ranges from 13 to 15. The direct digital synthesis circuit is thus capable of a higher level of precision than the residual time count. Therefore, to more precisely control the service clock frequency, destination node 100 looks at a number of samples of RTS values over a plurality of time windows to determine "interpolated" count values. The interpolated count values are used to calculate a value to write to the register of direct digital synthesis circuit 120 so as to produce a desired, more accurate frequency for the local service clock.

[0026] Direct digital synthesis circuit 120 provides this local service clock signal to buffer 114 and frame/line interface unit (frame/LIU) 118. Frame/LIU 118 provides an output signal such as a T1 output. The clock signal from direct digital synthesis circuit 120 controls the rate at which data is processed by buffer 114 and frame/LIU 118. Thus, if the frequency of the signal from direct digital synthesis circuit 120 is too low, the level of data in buffer 114 will increase and, if direct digital synthesis circuit 120 is not adjusted, buffer 114 could overflow and data could be lost. Thus, microcontroller 116 uses both RTS values and buffer fill data to control direct digital synthesis circuit 120.

[0027] Microcontroller 116 is programmed to generate a number that sets the frequency of direct digital synthesis circuit 120 based on RTS values processed by counting circuit 112 over a plurality of time periods. In another embodiment, microcontroller 116 uses an indication of the fill level of buffer 114 to further adjust the frequency of direct digital synthesis circuit 120. Advantageously, the use of microcontroller 116 to control direct digital synthesis circuit 120 based on received RTS values and buffer fill levels avoids the complexity of a closed, feedback loop which would need to be replicated for each port of the node. The techniques relating to RTS values and buffer fill levels are described in turn below.

[0028] Microcontroller 116 executes a process that effectively interpolates RTS values over a plurality of time periods to derive a number that is used to set the output of direct digital synthesis circuit 120. The following derivation is provided to assist in understanding the relationship between the RTS values monitored in a given time period and the number that is writable to the register, FREQ. REG., of direct digital synthesis circuit 120. First, RTS values are generated at source node 104 by running a counter with the network clock. This means that the value in the counter is incremented by 1 every cycle of the network clock. The value in the counter is "latched" or read out of the counter each time the service clock counter at the source node completes a count of Q, e.g., 3008 for T1 and E1 service. This physical relationship can be represented mathematically as shown in Equation 1:

$$TOTAL\ COUNT = f_{NET} \bullet T_S \bullet Q = f_{NET} \bullet \frac{Q}{f_S}$$

[0029] In Equation 1, the term f_{NET} refers to the frequency of the network clock, e.g., 2.43×10^6 for T1 and E1 service. T_S is the period of the service clock which is multiplied by a factor, Q. The period of Q service clocks represents the amount of time over which the counter is incremented and the frequency of the network clock represents the number of times per second that the counter is incremented. Thus, the product of these two quantities, f_{NET} and $T_S \cdot Q$, gives the value in the counter.

[0030] One underlying assumption in Equation 1 is that the counter has a sufficient number of bits to count the cycles of the network clock during the modified service clock period. However, a 4-bit counter is specified for RTS generation. Thus, when the counter reaches its highest value, i.e., 15 for a 4 bit counter, it wraps back around to zero and starts over again. At the end of Q periods of the service clock, there is a value stored in the counter. This value is the RTS value that is transmitted to destination node 100.

[0031] The counter used to generate RTS values is not reset when an RTS value is read out of the counter. For the next RTS value, there is a residual, usually non-zero, value in the counter. Thus, consecutive RTS values typically are not the same. However, for simplicity in the derivation, RTS values will be considered to be 16 minus the difference between consecutive readings of the counter, taking into consideration that the counter wraps back around to zero when it reaches 15 (e.g., the two's complement of the difference between the RTS values). For example, consecutive counter values of 14, 12, 10, 8, and 6 will be treated as five consecutive RTS values of 14.

[0032] Equation 1 can be modified to account for the wrapping nature of a P-bit counter as shown in Equation 2.

$$RTS = \left(\frac{f_{NET} \bullet \frac{Q}{f_S}}{2^P} \bullet NUMBER\ OF\ WRAPS \right) \_ 2^P$$

[0033] In Equation 2, the first term represents the number of times that a P-bit counter will roll over in generating an RTS value, plus a decimal residue. This term is a rational number, i.e., including a value to the right of the decimal. Thus, the difference of the two terms in the parenthesis gives a fractional value that is proportional to the value in the P-bit counter, the residue, when it was latched by the modified service clock. When this value is multiplied by 2^P, the result is the value of the 4-bit counter if the counter had been set to zero prior to count period. *NUMBER OF WRAPS* is the number of times that the 4-bit counter wraps for Q periods of the service clock. For T1 service, *NUMBER OF WRAPS* is 295 and for E1 service *NUMBER OF WRAPS* is 223.

[0034] Equation 2 can be manipulated to derive an equation for the service clock frequency, f_S, as a function of interpolated RTS values (RTS'), $f_S = F(RTS')$, as shown in Equation 3.

$$f_S = F(R'TS) = \frac{\frac{f_{NET} \bullet Q}{2^P}}{\frac{R'TS}{2^P} + NUMBER\ OF\ WRAPS}$$

[0035] Using Equation 3, destination node 100 can recover the service clock frequency of source node 104 using only the RTS values.

[0036] As described above, a direct digital synthesis circuit produces a signal with a frequency that is a fraction of a reference clock. This is represented in equation 4:

$$f = \frac{x}{2^n} \bullet f_{REF}$$

[0037] In equation 4, the value X represents the value that is written to the direct digital synthesis circuit to set the frequency of its output. The number n is the number of bits in the value to be written and f_{REF} is the frequency of the reference clock. Equation 4 can be solved for X in terms of the frequency of the service clock as shown in Equation 5:

$$X = \frac{2^n}{f_{REF}} F(R'TS)$$

[0038] In this equation, the desired frequency (*f*) has been replaced with the expression *F(RTS')*. Thus, Equation 5 provides a relationship between the number to be written to the register of direct digital synthesis circuit 120 that is based entirely on the RTS values.

[0039] Advantageously, microcontroller 116 processes RTS values over a plurality of time windows before adjusting the setting of direct digital synthesis circuit 120. For example, in one embodiment, microcontroller 116 processes RTS samples over four time windows, namely, time windows of 2, 20, 200 and 2000 seconds. For each time window, microcontroller 116 considers samples of RTS values to determine a value X to be written to the register of direct digital synthesis circuit 120. In determining this value, microcontroller 116 uses Equation 6 to determine an interpolated value of RTS, *RTS',* to be used in calculating the value of X with Equation 5 for each time window.

$$RTS' = \frac{Nx + My}{x + y}$$

[0040] Equation 6 represents a calculation of the mean RTS value when there are x RTS values of N (majority count) and y RTS values of M (minority count), with x greater than y. By processing a large number of samples, direct digital synthesis circuit 120 can be controlled to produce a reference clock with an accuracy of approximately 0.1 PPM, assuming that the network clock frequencies are the same at source node 104 and destination node 100.

[0041] In one embodiment, counting circuit 112 maintains three counters to provide microcontroller 116 with the RTS samples necessary to control DDS circuit 120. Each counter of counting circuit 120 corresponds to one of three expected RTS values (namely, 13, 14, and 15). The RTS values used by counting circuit 112 are calculated by using the two's complement of the difference between two consecutive RTS values. Microcontroller 116 reads the counters of counting circuit 112 every 2 seconds. Microcontroller 116 uses the read values to update the majority and minority counts for the current 2, 20, 200, and 2000 second windows for use in calculating the value to

control DDS circuit 120. The flow chart of Figure 2, described below, provides one process for using the majority and minority counts from the four time windows to generate an input to control DDS circuit 120.

[0042] Microcontroller 116 can implement Equation 5 without the need to use floating point calculations. For example, Equation 5 can be modified as follows for delivering T1 service with a reference clock that is 19.44 MHz:

$$X = \frac{48{,}128\_8{,}192\_4{,}096}{R'TS + 4{,}720}$$

[0043] By limiting the equation to integer values, floating point calculation can be avoided. In other embodiments, floating point calculations can be used when an appropriate processor is available.

[0044] Figure 2 is a flow chart of an embodiment of a process for setting a frequency of a local service clock according to the teachings of the present invention. The method of Figure 2 begins at block 200. This embodiment makes adjustments to a direct digital synthesis circuit based on monitored RTS values over four time windows although, in other embodiments, the number of time windows used may vary. The process calculates control values, labeled X_N with the subscript N corresponding to the number of seconds in a time window, for the direct digital synthesis circuit based on the RTS sample for each time window. Then, based on specified comparisons, the process selects one of the control values.

[0045] At block 202, the method determines whether the value X_2 for the two second window is within 0.2 parts per million (PPM) of the value X_{2000}. If not, then there has been sufficient change in the short term to adjust the direct digital synthesis circuit based on the value X_2 at block 204. If, however, the value for X_2 is within the specified limit of the value for X_{2000}, then the method proceeds to block 208.

[0046] At block 208, the method determines whether the value X_{20} for the twenty second window is within 20 parts per billion (PPB) of the value X_{2000}. If not, then there has been sufficient change to adjust the direct digital synthesis circuit based on the value X_{20} at block 210. If, however, the value for X_{20} is within the specified limit of the value for X_{2000}, then the method proceeds to block 212.

[0047] At block 212, the method determines whether the value X_{200} for the two hundred second window is within 2 parts per billion (PPB) of the value X_{2000}. If not, then there has been sufficient change to adjust the direct digital synthesis circuit based on the value X_{200} at block 214. If, however, the value for X_{200} is within the specified limit of the value for X_{2000}, then the method proceeds to block 216 and uses the value for X_{2000}.

[0048] The method ends at block 206.

[0049] As mentioned above, microcontroller 116 uses the average buffer fill level data to calculate a more optimum setting of direct digital synthesis circuit 120. It is noted that microcontroller 116 in another embodiment uses peak buffer fill level information as described below with respect to Figure 3. The peak buffer fill level information can be used alone or in combination with RTS values. Microcontroller 116 analyzes the buffer fill levels to evaluate three related aspects of the operation of destination node 100. First, microcontroller 116 uses the buffer fill level data to monitor the quality of the *RTS'* service clock recovery for a particular port described above. Further, microcontroller 116 uses the buffer fill level data across all ports to analyze for possible differences in the network clock at the source and destination nodes. Finally, microcontroller 116 uses the buffer fill level data to calculate Cell Delay Variation at a particular port of the destination node to determine an optimum level for that port. Each of these functions of microcontroller 116 are discussed, in turn, below.

[0050] Microcontroller 116 analyzes the buffer fill levels for each port of a destination node to assist in synchronizing the service clock at destination node 100 with the service clock at the source node for the port. In one embodiment, microcontroller 116 receives an update on the fill level of buffer 114 with the arrival of each new cell. If the service clock for a port of destination node 100 is substantially

synchronized with the service clock of source node 104, then the average fill level of buffer 114 should remain approximately constant over time since data packets are being generated at source node 104 and processed at destination node 100 at approximately the same speed. However, as mentioned, network 102 can introduce a variable delay for each packet it transports between source node 102 and destination node 100. This delay is typically referred to as "cell jitter" or "cell delay variation." To account for this cell delay variation, microcontroller 116 looks at the buffer fill level for a period of time before making an adjustment, if any, to the setting of direct digital synthesis circuit 120. For example, microcontroller 116 can average the buffer fill level data over a period of 10 seconds for T1 service.

[0051] With T1 service, if the service clock at the destination node is in error by -1 PPM, then the average buffer fill level for a port will increase by 1 byte in approximately 5.18 seconds. With this information, microcontroller 116 can determine an appropriate offset for direct digital synthesis circuit 120 so as to synchronize the local service clock with the service clock at source node 104. If the buffer fill level is increasing, then the service clock speed is increased so that packets are processed faster. If, however, the buffer fill level is decreasing then the service clock speed is decreased so that packets are processed slower.

[0052] Microcontroller 116 also analyzes the buffer fill level data over all ports on destination node 100 to detect an error in network clocks between source and destination nodes. When a significant correlation in drift of the buffer fill levels of all ports is detected, a correction can be added to the setting of direct digital synthesis circuit 120 to compensate for the difference in network clock frequencies. Such differences may be due, for example, to not having a common network reference clock at the source and destination nodes. Microcontroller 116 can use this correction factor for each port of destination node 100.

[0053] Finally, microcontroller 116 also analyzes the buffer fill level data to determine an optimal operating point for a buffer for a particular port. Microcontroller 116 uses the information provided by buffer 114 to determine cell-to-cell delay variation. Further, microcontroller 116 uses the extremes of cell delay variation to

determine peak-to-peak cell delay variation. With this information, microcontroller 116 can set the frequency of direct digital synthesis circuit 120 to achieve an average buffer fill level that accommodates the expected peak-to-peak cell delay variation, without adding excess cell delay.

[0054] Advantageously, by averaging the buffer fill level data for 10 seconds before adjusting the direct digital synthesis circuit in each of the above operations, the maximum wander component frequency that can be found on the output of the local service clock is 0.05 Hz.

III. Adaptive Clock Recovery

[0055] Figure 3 is a block diagram of an illustrative embodiment of the present invention. A source node 300 sends data packets through an ATM network 303 to a destination node 305. A service clock 301 regulates the rate at which the source node 300 transmits data to the destination node 305.

[0056] Destination node 305 receives and processes data packets from source node 300. Advantageously, destination node 305 uses an adaptive clock recovery scheme that monitors a peak buffer fill level to lock a local oscillator of destination node 305 with the frequency of service clock 301 in a manner that meets system wander limits.

[0057] Destination node 305 includes reassembler 307. Reassembler 307 receives the data packets from the source node 300 and places the data packets in proper sequence. Reassembler 307 is coupled to buffer 309. Buffer 309 is coupled to framer/Line Interface Unit (L.I.U.) 311. Buffer 309 receives the data packets from the reassembler 307 then passes the data packets, sequentially to the framer/L.I.U. 311. Framer/L.I.U. 311 receives the data packets from the buffer 309 and passes the data from the data packets according to the original format. Any data format can be used by the framer/L.I.U. 311. In one embodiment, the data is formatted for transmission on a T1 line.

[0058] The reading and writing of data to and from buffer 309 is controlled by addresses label W ADDR (□write address") and R ADDR ("read address"). The write address identifies where the most recent data was written to buffer 309 from

reassembler 307. The read address identifies where the most recent data was read from buffer 309 by framer/L.I.U. 311. Thus, the difference between these two addresses is indicative of a fill level of the buffer.

[0059] As data is processed by destination node 305, read and write addresses are compared by peak fill level detector 313 and the difference is stored in a register. The peak fill level detector 313 may be implemented in a Field Programmable Gate Array ("FPGA") or may be performed by the microprocesssor, if every buffer fill sample is made available. Peak fill level detector 313 continues to compare read and write addresses for a period of time, e.g., one to two seconds, to check if the new buffer fill level based on the difference between the read address and write address is greater than the current value in the register. If so, the peak fill level detector 313 replaces the value in the register with the current buffer fill level. Ultimately, at the end of the period of time, the register contains a peak fill level for the buffer 309.

[0060] At the end of the period of time, a processor 315 reads the register in the peak fill level detector 313 to get the peak fill number. Processor 315 clears the register to zero. Peak fill level detector 313 then repeats the process to obtain further peak fill numbers for further time periods.

[0061] Processor 315 uses the peak fill number from the register and a clock control algorithm to adjust, as necessary, the input to a numerically controlled oscillator 317. The output of the numerically controlled oscillator 317 is the local clock of destination node 305 and, by means of an adaptive algorithm, is locked to service clock 301. This process excludes data for loop control that has been corrupted by Cell Transfer Delay Variation.

[0062] Numerically controlled oscillator 317 establishes an output frequency based on a number provided by processor 315 and the frequency of reference clock 319. In one embodiment, numerically controlled oscillator 317 comprises a Direct Digital Synthesis ("DDS") integrated circuit available from Analog Devices. Further, reference clock 319 is a clock with an accuracy of a stratum 1, 2, or 3E clock. The frequency of reference clock 319, is thus, assumed to be fixed at a selected level, e.g., 19.44 MHz. Thus, to control the frequency of the signal output by numerically controlled oscillator

317, it is only necessary to determine the appropriate numeric value to provide to numerically controlled oscillator 317. This process results in a recovered clock that contains no jitter.

[0063] Source node 300 can transmit signals to destination node 305 in a number of different standard formats. For example, these formats include, but are not limited to, DS1, E1, E3, and DS3. Each of these formats has a nominal frequency associated with it. A number for numerically controlled oscillator 317 that will achieve a target frequency is calculated according to equation 1:

$$Number = \frac{Target\ Frequency\ x\ 2^{32}}{Reference\ Frequency}$$

[0064] In Equation 1, the *Target Frequency* is the nominal frequency for the selected service, e.g., 1.544 MHz for DS1, 2.048 MHz for E1, 17.184 MHz for E3 and 22.368 MHz for DS3 service. The reference frequency is the frequency of reference clock 319. The number, 2^{32}, represents the highest value of the 32 bit number that can be applied to numerically controlled oscillator 317. Essentially, the number applied to numerically controlled oscillator 317 sets a ratio between the frequency of the output and the reference clock 319. In the case of DS1 and E1, a 19.44MHz reference clock can be used. Similarly, a 77.76 MHz reference frequency clock can be used for E3 and DS3 service, but a 2x clock multiplier follows the numerically controlled oscillator output to achieve the required 34.368MHz for E3, and 44.736MHz for DS3. Newer DDS circuits can produce the required output frequency directly. With these values, the number that achieves these nominal frequencies for the identified services are as follows:

Service	Number
DS1	341,122,916.925

E1	452,473,920.896
E3	949,134,748.129
DS3	1,234,465,901.19

[0065] Since only integer numbers can be provided, the next highest and next lowest integer values are used. The frequencies that result from these values can be calculated according to Equation 2:

$$Frequency = \frac{Number\ x\ Reference\ Frequency}{2^{32}}$$

[0066] This results in the following frequencies for each service:

Service	Frequency (High)	Frequency (Low)	Difference
DS1	1,544,000.00034 Hz	1,543,999.99581 Hz	0.00453 Hz
E1	2,048,000.00047 Hz	2,047,999.9959 Hz	0.00457 Hz
E3	17,184,000.0158 Hz	17,183,999.9977 Hz	0.0181 Hz *
DS3	22,368,000.0146 Hz	22,367,999.9965 Hz	0.0181 Hz *

[0067] From the difference values, the effect of a one bit change in the value of the number can be calculated for each service:

[0068] For each service, the effect, in nanoseconds-per-second is as follows:

Service	Effect
DS1	2.9339
E1	2.231
E3	1.053 *
DS3	0.8092 *

[0069] * The difference frequency in equation (3) must be 2 times the table value, because the DDS output frequencies are doubled to achieve the required frequencies.

[0070] Essentially, processor 315 can calculate the rate of change of the buffer fill level in terms of a nanosecond-per-second value. Based on this value, processor 315 can determine an amount by which to adjust the number provided to numerically controlled oscillator 317 to compensate for the change in the buffers fill level using the above numbers for the selected service.

[0071] Processor 315 uses a clock control algorithm to acquire frequency and phase lock and to track phase of the service clock 301 by the destination node 305. The clock control algorithm has three different functions: 1. frequency acquisition, 2. phase acquisition, and 3. phase track.

[0072] Before transmission of a signal by source node 300, the buffer fill level at the destination node 305 reduces to zero, because the data is clocked out of the buffer by the numerically controlled oscillator 317, but no valid data is being written into the buffer from the source node 300. The numerically controlled oscillator 317 is then set to a frequency equal to the lowest frequency allowed by specification for service clock 301. When data is received at the buffer 309, the buffer fill level changes. This change is calculated as a nanosecond per second change. Based on this value, a change in the number to the numerically controlled oscillator 317 can be calculated and written to reduce the frequency offset to zero.

[0073] Phase uncertainty is introduced at destination node 305 since buffer 309 is configured byte-wide. This introduces an 8-unit interval (UI) uncertainty in the buffer fill level. By the time frequency lock is achieved, there is typically a higher fill level than target fill level because the numerically controlled oscillator 317 was set to read data from the buffer 309 at a slower rate than data written to the buffer 309, in order to accumulate data to calculate frequency offset. After frequency acquisition, buffer 309 fill level is reduced to target level at a controlled rate to avoid data loss. This is achieved by increasing the frequency of the numerically controlled oscillator 317 to a slightly higher value than the frequency of service clock 301. Once the buffer fill level is reduced to the selected target level, the frequency of the numerically controlled oscillator 317 is lowered slightly below the frequency lock value in order to transition between a target byte and a next higher byte. When this boundary is located, the number of the numerically controlled oscillator 317 will be changed to the zero frequency offset number. At this point, a phase detector with bit level resolution is realized.

[0074] The last function of the clock algorithm is the phase track where a number from the numerically controlled oscillator 317 is adjusted by one digit or more digits as required to maintain phase alignment at the byte boundary described.

[0075] In one embodiment, the clock recovery mechanism of Figure 3 is also implemented over a plurality of time periods as described above with respect to Figure 2. In this embodiment, the method of Figure 2 begins at block 200. This embodiment makes adjustments to a direct digital synthesis circuit based on peak buffer fill levels over four time windows although, in other embodiments, the number of time windows used may vary. The process calculates control values, labeled X_N with the subscript N corresponding to the number of seconds in a time window, for the direct digital synthesis circuit based on the peak fill levels for each time window. Then, based on specified comparisons, the process selects one of the control values.

[0076] At block 202, the method determines whether the value X_2 for the two second window is within 0.2 parts per million (PPM) of the value X_{2000}. If not, then there has been sufficient change in the short term to adjust the direct digital synthesis

circuit based on the value X_2 at block 204. If, however, the value for X_2 is within the specified limit of the value for X_{2000}, then the method proceeds to block 208.

[0077] At block 208, the method determines whether the value X_{20} for the twenty second window is within 20 parts per billion (PPB) of the value X_{2000}. If not, then there has been sufficient change to adjust the direct digital synthesis circuit based on the value X_{20} at block 210. If, however, the value for X_{20} is within the specified limit of the value for X_{2000}, then the method proceeds to block 212.

[0078] At block 212, the method determines whether the value X_{200} for the two hundred second window is within 2 parts per billion (PPB) of the value X_{2000}. If not, then there has been sufficient change to adjust the direct digital synthesis circuit based on the value X_{200} at block 214. If, however, the value for X_{200} is within the specified limit of the value for X_{2000}, then the method proceeds to block 216 and uses the value for X_{2000}.

[0079] The method ends at block 206.

Conclusion

[0080] Although specific embodiments have been illustrated and described herein, it will be appreciated by those of ordinary skill in the art that any arrangement which is calculated to achieve the same purpose may be substituted for the specific embodiment shown. This application is intended to cover any adaptations or variations of the present invention. For example, the service clock recovery technique can be used for other services, e.g., E1 and other conventional continuous bit rate services, and this application is not limited to use with T1 service. In other embodiments, RTS values are used in conjunction with peak buffer fill levels. In other embodiments, the number of time periods used to monitor the control signals for the recovered clock varies from the 2, 20, 200, and 2000 second time periods described here. Further, in other embodiments, the microcontroller uses other appropriate comparisons, e.g., different acceptable ranges, among the calculations for the various time windows to select the control signal used to control the local clock.